Tile Complexity of Approximate Squares and Lower Bounds for Arbitrary Shapes

Harish Chandran[*] harish@cs.duke.edu

Nikhil Gopalkrishnan[*] nikhil@cs.duke.edu

John Reif[*] reif@cs.duke.edu

Abstract

We consider the problem of building approximate squares in the standard Tile Assembly Model. Given any $\epsilon \in (0, \frac{1}{4}]$ we show how to construct squares whose sides are within $(1 \pm \epsilon)N$ of any given positive integer N using $\mathcal{O}\left(\frac{\log \frac{1}{\epsilon}}{\log\log \frac{1}{\epsilon}} + \frac{\log\log \epsilon N}{\log\log\log \epsilon N}\right)$ tile types. We prove a matching lower bound by showing that $\Omega\left(\frac{\log \frac{1}{\epsilon}}{\log\log \frac{1}{\epsilon}} + \frac{\log\log \epsilon N}{\log\log\log \epsilon N}\right)$ tile types are necessary almost always to build squares of required approximate dimensions. In comparison, the optimal construction for a square of side exactly N uses $\mathcal{O}\left(\frac{\log N}{\log\log N}\right)$ tile types. Our constructions can get highly accurate squares using a very few number of tile types and are feasible starting from values of N that are orders of magnitude smaller than the best comparable constructions previously suggested. We also give lower bounds for the number of tile types required to assemble any arbitrary shape by associating shapes with binary numbers via a deterministic, computable procedure.

1 Introduction

Self-assembly is a fundamental pervasive natural phenomenon that gives rise to complex structures and functions. It describes processes in which a disordered system of pre-existing components form organized structures as a consequence of specific, local interactions among the components themselves, without any external direction. In its most complex form, self-assembly encompasses the processes involved in growth and reproduction of higher order life. A simpler example of self-assembly is the orderly growth of crystals. In the laboratory, self-assembly techniques have produced increasingly complex structures (see Park et al. (2005); Andersen et al. (2009); Rothemund (2006); Douglas et al. (2009); Dietz et al. (2009) for some illustrative examples) and dynamical systems (see Dirks and Pierce (2004); Yin et al. (2004); Zhang et al. (2007); Yin et al. (2008)). While a surprising degree of control is exhibited in self-assembled systems in nature, a certain degree of flexibility is inherent in many systems. This suggests that self-assembly processes need not exactly match the desired outcome and approximate notions may be sufficient. The roots of attempts to model and study self-assembly begin with the study of tilings. In this paper we explore the notion of approximation for tilings of a square and attempt to gain insights into the minimum tile set needed to construct arbitrary shapes by supplying two general lower bounds for tile complexity of arbitrary shapes.

1.1 Tiling Systems that Model Self-assembly

A *Wang tile* (1961) is an oriented unit square with a pad associated with each side. Any two tiles with the same pads on all corresponding sides are said to be of the same *tile type*. Given a finite set S of Wang tiles types, a valid arrangement of S on a planar unit square grid consists of copies of Wang tiles from the set S such that abutting pads of all pairs of neighboring tiles match. The *tiling* or *domino* problem for a set of Wang tiles is: can tiles from S (chosen with replacement) be arranged to cover the entire planar grid? Berger (1966) proved the undecidability

[*]Department of Computer Science, Duke University, Durham, North Carolina, USA.

of the tiling problem by reducing the halting problem to it. Robinson (1971) gave an alternative proof involving a simulation of any single tape deterministic Turing Machine by some Wang tiling system. Garey and Johnson (1981) and Lewis and Papadimitriou (1981) proved that the problem of tiling a finite rectangle is **NP**-complete. These results paved the way for Wang tiling systems to be used for computation. But, Wang tilings do not model coordinated growth and hence do not describe complex self-assembly processes. Winfree et al. (1996) extended Wang tilings to the Tile Assembly Model (TAM) with a view to model self-assembly processes, laying a theoretical foundation for a form of DNA based computation (see Rothemund and Winfree (2000) for details of the model), in particular, molecular computation via assembly of DNA lattices with tiles in the form of DNA motifs.

Rothemund and Winfree (2000) define *tile complexity* of a shape as the minimum number of tile types for assembling that shape. Tile complexity, apart from capturing the information complexity of shapes, is also important as there exist fundamental limits on the number of tile types one can design using DNA sequences of fixed length. Various ingenious constructions for shapes like squares (see Adleman et al. (2001)), rectangles and computations like counting (see Barish et al. (2005)) etc. exist in this model. Lower bounds on tile set complexity have also been shown for various shapes but no non-trivial lower bounds have been reported for arbitrary shapes.

1.2 Outline of Results

A canonical problem in TAM and its extensions is the assembly of squares from unit sized square tiles. This simple but fundamental question in self-assembly has helped formulate notions of tile complexity and running time. Becker et al. (2006) and later Kao and Schweller (2008) used this question to introduce notions of probabilistic assembly and approximate shapes. Kao and Schweller achieve, with arbitrarily high probability, squares of required approximate dimensions using $O(1)$ tile types in an extension of TAM where tile attachments are probabilistic based on their relative concentrations. Doty (2009) extended this result to achieve, with arbitrarily high probability, squares of required exact dimensions using $O(1)$ tile types in the same model. Only a constant number of tile types are required because all the information content of the square is encoded in the tile concentration assignment. Thus, the number of tile types is not a measure of the information complexity of the shape.

We wish to separate the notion of approximate shapes from probabilistic assembly and consider them independently. Chandran et al. (2009) introduced notions of probabilistic assembly that encode concentrations implicitly in tile set specification. Section 5 deals with purely deterministic assembly of approximate squares in the standard TAM where the minimum number of tile types for forming an approximate square is closely related to the descriptional complexity of the shape. Given any $\epsilon \in (0, \frac{1}{4}]$ we show in section 5.1 how to construct squares whose sides are within $(1 \pm \epsilon)N$ of any given positive integer N using $\mathcal{O}\left(\frac{\log \frac{1}{\epsilon}}{\log\log \frac{1}{\epsilon}} + \frac{\log\log \epsilon N}{\log\log\log \epsilon N}\right)$ tile types. We prove a matching lower bound in section 5.3 by showing that $\Omega\left(\frac{\log \frac{1}{\epsilon}}{\log\log \frac{1}{\epsilon}} + \frac{\log\log \epsilon N}{\log\log\log \epsilon N}\right)$ tile types are necessary almost always to build squares of required approximate dimensions. In comparison, the optimal construction for a square of side exactly N uses $\mathcal{O}\left(\frac{\log N}{\log\log N}\right)$ tile types (see Adleman et al. (2001)).

Adleman et al. (2002) defines the Minimum Tile Set problem as finding the smallest tile set that uniquely produces a given shape. This problem has been well studied for squares but the general question for arbitrary shapes[1] has not received as much attention. This problem might be seen as corresponding to the question of finding the smallest program that outputs a binary string. While finding the smallest program that outputs a given string is uncomputable, the Minimum Tile Set Problem is **NP**-complete (see Adleman et al. (2002)). Thus finding the smallest tile set for an arbitrary shape is not tractable. In such a situation, a general lower bound on

[1]Note that any arbitrary shape is connected as it is a terminal configuration grown from a seed tile.

the minimum tile set required for an arbitrary shape may prove useful. An alternate approach was developed by Soloveichik and Winfree (2007) where using a notion of shapes independent of scale, the authors were able to bound the minimal tile set required for assembling a shape in terms of the Kolmogorov complexity of the shape. We do not consider shapes independent of scale. In section 6 we give lower bounds for the number of tile types required to assemble any arbitrary shape by associating shapes with binary numbers via a deterministic, computable procedure.

2 Related Work in Constructing Approximate Squares

Various techniques have been suggested previously to build approximate squares in tile assembly models. In contrast to our work, all these involve working in modified domains of the abstract Tile Assembly Model (TAM) of Winfree (1995). In the *temperature programming* modification, squares of arbitrary size are achieved by Kao and Schweller (2006) using a constant number of tile types by affecting a sequence of temperature changes. In the *staged assembly* modification by Demaine et al. (2007), the assembly process is performed in a sequence of stages allowing construction of arbitrary shapes using a constant number of tile types. In contrast to our work, in both these models the computational complexity of the target shape is encoded at least partly within the sequence of operations performed on sets of tiles as opposed to encoding all complexity purely in some attribute of the tile set itself.

In the *concentration programming* modification, the assembly is not locally deterministic as more than one tile type can compete for binding at the same location in an assembly and the relative probabilities of attachment are governed by tile type concentrations in the solution. Programming the relative concentrations of tile types controls the size and shape of assemblies created. Among the drawbacks of this model (in addition to the usual drawbacks of TAM): depletion of concentration of tiles as they attach to the growing assembly are usually not modeled and arbitrarily precise concentrations are often required. In the full version of his paper Doty (2009) discusses these problems and suggests methods to get around them. Becker et al. (2006) suggested using concentration programming to build squares of expected size N for all N using just 5 tile types. However, each square size is geometrically distributed about its expectation N and hence has high variance. Kao and Schweller (2008) through an ingenious use of concentration programming gave tile sets that for any precision ϵ, certainty δ and sufficiently large integer N achieve, with probability at least $1-\delta$, squares of size L, where $(1-\epsilon)N \leq L \leq (1+\epsilon)N$, using a constant number of tile types. However, for a $(0.01, 0.01)$ approximation their constructions are only proven to work, according to their analysis, for squares of size $N \geq 10^{13}$. Even for tiles as small as 1 nanometer across, $N \geq 10^{13}$ works out to 10 kilometer size squares. Doty (2009) significantly improved this result, achieving exact squares, with arbitrarily high probability $1-\delta$, of size N for sufficiently large N. For $\delta = 0.01$ certainty his constructions are proven to work, according to his analysis, for squares of size $N \geq 10^7$, which for tiles of size 1 nanometer works out to 1 centimeter. In simulations, he suggests that lower values of N may suffice. The number of tile types used in his construction for a $\delta = 0.01$ certainty is ≈ 5000.

For an accuracy $\epsilon = 0.01$, our constructions are proven to work for all $N \geq 13130$. For tiles of size 1 nanometer this works out to about 13 micrometers. The number of tile types used to achieve a square of size 10^7 at an accuracy of $\epsilon = 0.01$ is just 58. If we increase our accuracy to $\epsilon = 0.001$, we still need only 66 tile types to construct squares of size 10^7. Thus, we can get highly accurate squares using a very few number of tile types and our constructions are feasible starting from values of N that are orders of magnitude smaller than the best comparable constructions previously demonstrated.

3 Tile Assembly Model

This section describes the abstract Tile Assembly Model (TAM). For a complete description of the model see Rothemund and Winfree (2000). Readers familiar with the TAM may skip this section. Consider the two-dimensional grid of integers $\mathbb{Z} \times \mathbb{Z}$ on which our tiles lay. The directions

$\mathfrak{D} = \{\text{North}, \text{South}, \text{East}, \text{West}\}$ are functions from $\mathbb{Z} \times \mathbb{Z}$ to $\mathbb{Z} \times \mathbb{Z}$, with $\text{North}(x, y) = (x, y + 1)$, $\text{South}(x, y) = (x, y - 1)$, $\text{East}(x, y) = (x + 1, y)$ and $\text{West}(x, y) = (x - 1, y)$. We say that (x, y) and (x', y') are neighbors if $(x', y') \in \{\text{North}(x, y), \text{South}(x, y), \text{East}(x, y), \text{West}(x, y)\}$. Note that $\text{North}^{-1} = \text{South}$, $\text{East}^{-1} = \text{West}$ and vice versa. $\mathbb{N}$ is the set of natural numbers.

A *Wang tile* over the finite set of distinct *pads* Σ is a unit square whose four sides have pads (not necessarily distinct) from the set Σ. Formally, a tile t is an ordered pair of pads $(N_t, E_t, S_t, W_t) \in \Sigma^4$ indicating pad types on the North, East, South and West sides respectively. Note that tiles are not necessarily isomorphic under in place rotation or reflection. For each tile t, we define $\text{pad}_{\text{North}}(t) = N_t$, $\text{pad}_{\text{South}}(t) = S_t$, $\text{pad}_{\text{East}}(t) = E_t$ and $\text{pad}_{\text{West}}(t) = W_t$. Σ contains a special *null pad*, denoted by ϕ. The *empty* tile (ϕ, ϕ, ϕ, ϕ) represents the absence of any tile. Pads determine when two tiles attach. A function $g : \Sigma \times \Sigma \to \mathbb{N}$ is a *pad strength function* if it satisfies $\forall x, y \in \Sigma,\ g(x, y) = g(y, x)$ and $g(\phi, x) = 0$.

A *tiling system*, $\mathbb{T}$, is a tuple $\langle T, S, g, \tau \rangle$ where T containing the empty tile is the finite set of tile types, $S \subset T$ is the set of *seed* tiles, g is the pad strength function and τ is a global thermodynamic parameter. A *configuration* of T is a function $A : \mathbb{Z} \times \mathbb{Z} \to T$ with $A(0, 0) = s$ for some $s \in S$. For $D \in \mathfrak{D}$ we say the tiles at (x, y) and $D(x, y)$ *interact* with binding strength $g(\text{pad}_D(A(x, y)), \text{pad}_{D^{-1}}(A(D(x, y))))$. For all $s \in S$ a *start* configuration start_s is given by $\text{start}_s(0, 0) = s$ and $\text{start}_s(x, y) = \text{empty}$ otherwise. A configuration A is called *τ-stable* iff $\forall A(x, y) \neq \mathit{empty}$: $\sum_{D \in \mathfrak{D}} g(\text{pad}_D(A(x, y)), \text{pad}_{D^{-1}}(A(D(x, y)))) \geq \tau$. In addition any start configuration is also τ-stable. A tile $t \in T$ is said to τ-stably attach to a configuration A at position (x, y) iff $A(x, y) = \mathit{empty}$ and $\sum_{D \in \mathfrak{D}} g(\text{pad}_D(t), \text{pad}_{D^{-1}}(A(D(x, y)))) \geq \tau$. *Self-assembly* is defined by a relation between configurations, $A \to B$, if there exists a tile $t \in T$ that τ-stably attaches to A to form B. We define $A \xrightarrow{*} B$ as the transitive closure of $\to$ and say B is *derived* from A. A configuration B is *produced* if $\text{start}_s \xrightarrow{*} B$ for some $s \in S$. A configuration is *terminal* if it is produced from start_s for some $s \in S$ and no configuration can be derived from it. $\text{Term}(\mathbb{T})$ is the set of terminal configurations of $\mathbb{T}$. In TAM, a terminal configuration is thought of as the output of a tiling system given a seed tile $s \in S$. We restrict ourselves to tile systems with a unique seed tile. Also, our pad strength function is restricted to be *diagonal*, i.e., $\forall x \neq y\ :\ g(x, y) = 0$. Finally, we ask that there be a unique terminal configuration, $\text{Term}(\mathbb{T}) = \{C\}$. Note that different attachment orders are allowed as long as they produce the same terminal configuration. DNA nanostructures can physically realize TAM as shown by Winfree et al. (1998) with the DX tile and LaBean et al. (2000) with the TX tile. Like the square tile in TAM, the DX and TX have *pads* that specify their interaction with other tiles. The pads are DNA sequences that attach via hybridization of complimentary nucleotides. Mao et al. (2000) performed a laboratory demonstration of computation via tile assembly using TX tiles. Yan et al. (2003) performed parallel XOR computation in the test-tube using Winfree's DX tile. Other simple computations have also been demonstrated. However, large and more complex computations are beset by errors and error correction remains a challenge towards general computing using DNA tiles.

4 Constructing Squares in the Tile Assembly Model

A terminal configuration C is called an $N \times N$ square iff there exists a position (x_0, y_0) such that $\forall x, y \in \mathbb{Z}$ satisfying $x_0 \leq x < x_0 + N$ and $y_0 \leq y < y_0 + N : C(x, y) \neq \mathit{empty}$. For all other positions, $C(x, y) = \mathit{empty}$. Self-assembly of squares at $\tau = 1$ is relatively uninteresting (see Rothemund and Winfree (2000) and Doty et al. (2009)) and hence we consider only $\tau \geq 2$ systems in this work.

Rothemund and Winfree (2000) demonstrated a construction for forming $N \times N$ squares for any given N using only $\mathcal{O}(\log N)$ tile types. The tiles self-assemble into a fixed width binary counter in the shape of a long rectangle using $\lceil \log N \rceil$ tile types. A fixed set of distinct *filler* tiles are then used to complete the rectangle into a square of the desired size. They further proved, using information theoretic arguments, that any construction for forming $N \times N$ squares needs $\Omega(\frac{\log N}{\log \log N})$ tile types for almost all N. This lower bound was achieved by Adleman (2000) who demonstrated a construction using $\mathcal{O}(\frac{\log N}{\log \log N})$ tile types for any given N using a more efficient counter. These results show that the program size complexity of self-assembled squares are closely related to the minimum tile set used to construct them.

Various modifications and additions to the tile assembly model have been proposed (see Aggarwal et al. (2004); Demaine et al. (2007); Becker et al. (2006)). The minimum tile set required to construct squares in these models have been studied, with dramatic improvements in certain cases. However, the relationship between number of tile types and program size complexity of the square is not preserved in most of these models. We restrict ourselves to the standard tile assembly model of Rothemund and Winfree (2000) in this work.

4.1 Self-Assembly of a Binary Counter

Fig. 1 illustrates a self-assembled infinite binary counter at $\tau = 2$, first described in Rothemund and Winfree (2000). The system consists of seven tiles. L, S (the seed tile) and R are the frame tiles which delimit the counter. L and R each bind to S and with themselves with strength 2 pads, indicated by a dark grey strip on their sides. They do not appear anywhere else in the counter. Two tiles with face label[†] 0 and two with face label 1 perform binary counting. Thus the face label of the tile at the $(i+1)^{\text{th}}$ row (from the bottom) and $(j+1)^{\text{th}}$ column (from the right) is the j^{th} least significant bit of the counter after i steps of counting, assuming counting initiated at zero. They have strength one pads on all four sides. One of the pad labels from $\{0, 1\}$ occur on the North and South sides and indicate bit value. One of the pad labels from $\{c, n\}$ occur on the East and West sides and indicate presence (c) or absence (n) of carry. The addition of the 0 and 1 tiles to the growing assembly occurs via binding interactions on its East and South sides. Note that both these pads must match for binding to take place at $\tau = 2$. Each of the 0 and 1 tiles are designed such that the sum of the bits represented by the South and East pads is propagated on the North pad as a bit value (0 or 1) and on the East pad as the carry (n or c). A fixed width version of this binary counter is achieved with minor modifications as discussed in the next section.

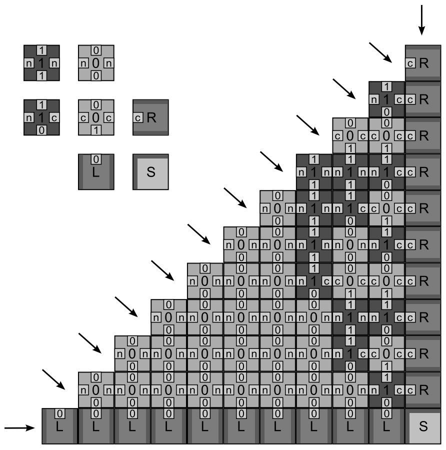

Figure 1: Counter of Rothemund and Winfree using Seven Tile Types

4.2 Squares Using a Finite Counter

Fig. 2 (i) generically illustrates Rothemund and Winfree (2000) construction for a $N \times N$ square using $\mathcal{O}(\log N)$ tile types. At the heart of the construction is a binary counter similar to the one described in Section 4.1. The chief difference in the two counters is that the new counter can start counting from any given n-bit binary number to a finite number and halts. This introduces additional complexity in the tile system necessitating additional tile types. The counter, a thin rectangle, is then extended into a square of the required size using diagonal building and filler tiles.

The system consists of a set of input tiles specific to the target dimension of the square and twenty three other tiles independent of the dimension. Recall that the thermodynamic parameter τ is set to 2. The seed row tiles (input tiles), $\Theta(\log N)$ in number, account for almost all of the descriptional complexity of the shape. They initiate the assembly by encoding the binary number to start counting from. The leftmost (rightmost) column of the counter is built using border tiles with no binding attachments to other counter tiles on the West (East) side thus restricting the counter to a finite width. A pair of rows of the counter encode the same binary number, where the top row copies the bottom row. The copy row conserves the fixed width nature of the counter while at the same time propagating the appropriate carry bit. This copying is achieved by two

[†]The face labels are identical to the North pad type and do not play any role in tile binding. They merely depict the bits of the counter.

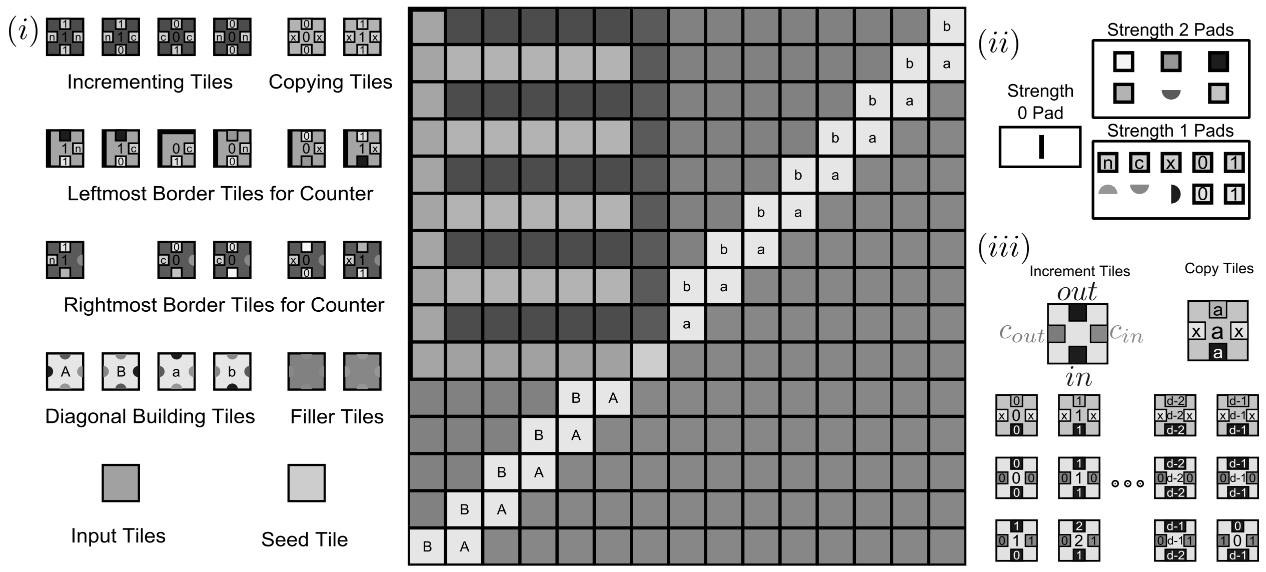

Figure 2: (i) $N \times N$ Square using $\mathcal{O}(\log N)$ tile types. (ii) Pads for $N \times N$ Square using $\mathcal{O}(\log N)$ tile types. (iii) Increment and Copy Tiles for Base d. The border tiles are not shown. The number of tile types is $\Theta(d)$.

copy tiles while the increment operation is implemented by four increment tiles analogous to those in Fig. 1. The counter halts when the leftmost bit rolls over from 1 to 0, indicated by a tile with a null pad (ϕ) on its North side. There are two pairs of diagonal building tiles which form a staggered diagonal from the seed tile. The rest of the square is completed with two filler tiles giving a $N \times N$ square for any given N using $\mathcal{O}(\log N)$ tile types. For more details see Rothemund and Winfree (2000).

The tile complexity for assembling an $N \times N$ square was reduced to $\mathcal{O}(\frac{\log N}{\log \log N})$ by Adleman et al. (2001), asymptotically matching the information theoretic almost always lower bound of $\Omega(\frac{\log N}{\log \log N})$. Analogous to the Rothemund and Winfree (2000) construction, a counter is used to assemble a thin rectangle that is extended to a square. The difference being, counting is now performed in a higher base which reduces the number of tile types required to form the seed row. Recall that the seed row needs to be composed of distinct tile types which accounts for most of the tile complexity of the system. Fig. 2 (iii) illustrates $\Theta(d)$ tile types for fixed width counting in base d, apart from the seed row. By choosing $d = \lceil \frac{\log N}{\log \log N} \rceil$ we can assemble $N \times N$ squares for any given N using $\mathcal{O}(\frac{\log N}{\log \log N})$ tile types. For more details see Adleman et al. (2001). We shall use this basic construction in the next section for building squares of dimension close to a given target N using exponentially lower number of tile types.

5 Constructing ϵ-Approximate Squares

Optimally concise tile sets for self-assembly of squares of exact size N have been well studied and the lower bound achieved, leaving no more room for improvement beyond constant factors. However, if we are allowed some error in side length, say by an additive fractional factor ϵ ($\pm\epsilon N$), we can design significantly more concise tile sets for squares. We describe such tile sets in this section. Our tile sets satisfy exactly the rules of the Tile Assembly Model of Rothemund and Winfree (2000). In particular, each tile set is diagonal and produces a unique terminal configuration in the shape of a square. Thus, the improved tile complexity we achieve can be compared with the optimal tile set for exactly sized self-assembled squares and the difference attributed precisely to the notion of approximation introduced by us.

5.1 Construction

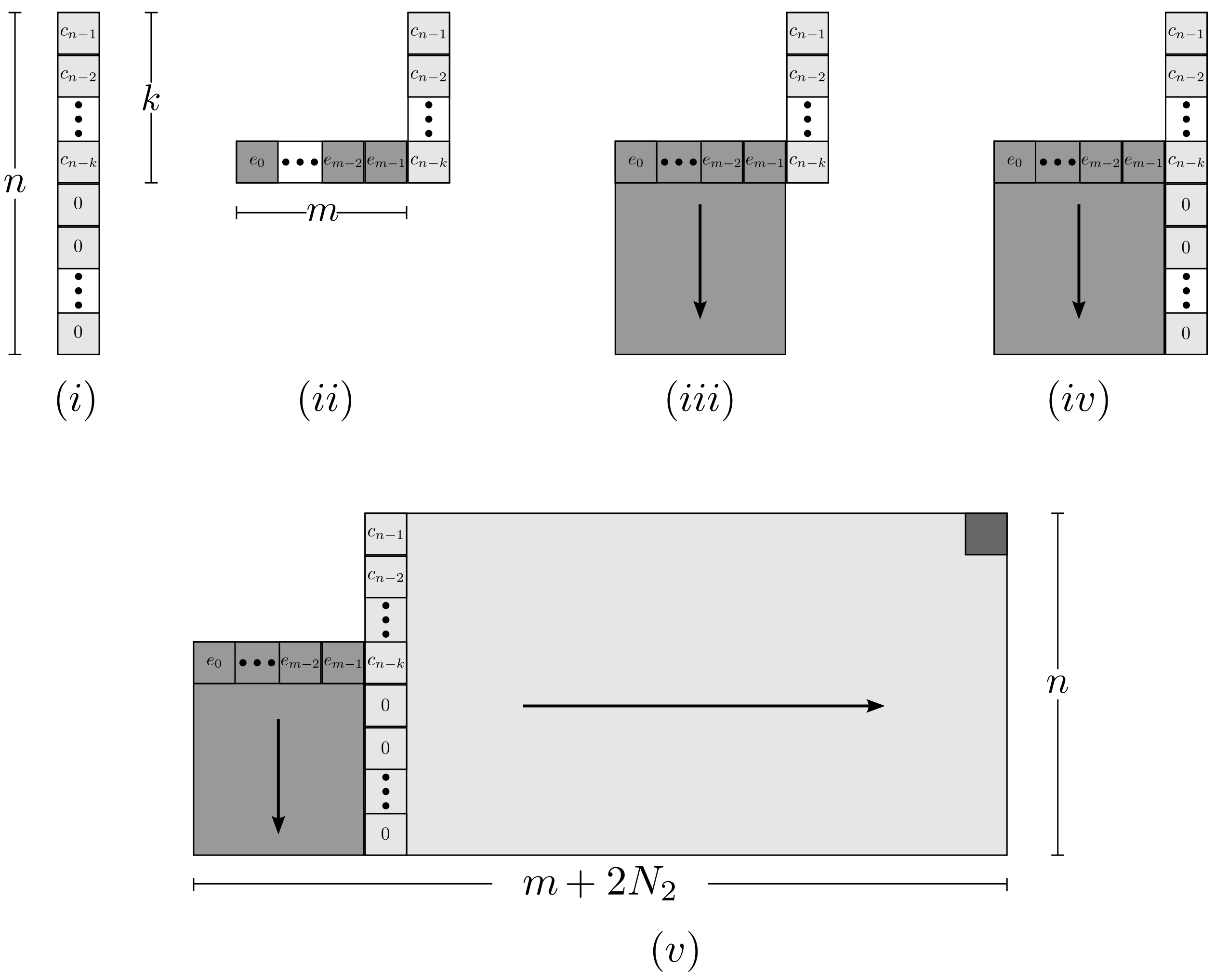

Figure 3: Components of the construction: (i) *Seed column* for the major counter. (ii) L-shaped seed assembly. (iii) Assembly of the minor counter. (iv) Completing the *seed column* using the 0 tile type. (v) Assembly of the major counter.

An $L \times L$ square is called an ϵ-*approximation* of an $N \times N$ square iff $(1-\epsilon)N \leq L \leq (1+\epsilon)N$. Since the size of any self-assembled square is integral, the error term ϵN vanishes for $N < \frac{1}{\epsilon}$ and hence we only consider $N \geq \frac{1}{\epsilon}$. We now describe our construction for an ϵ-approximation of an $N \times N$ square using $\mathcal{O}\left(\frac{\log \frac{1}{\epsilon}}{\log\log \frac{1}{\epsilon}} + \frac{\log\log \epsilon N}{\log\log\log \epsilon N}\right)$ tile types. Let d be a carefully chosen integer that we shall use as a base for a counter assembly described later. Given a number N, we construct a square of size L with $(1-\epsilon)N \leq L \leq (1+\epsilon)N$ for all sufficiently large N. Let $N_1 = \lfloor (N - \lfloor \log_d N \rfloor)/2 \rfloor$. Consider the encoding of N_1 in base d (d-ary encoding): $b_{n-1}b_{n-2}\ldots b_0$ where $n = \lfloor \log_d N_1 \rfloor + 1$. From N_1 we derive N_2 by setting all but the left most $k = \lceil \log_d \frac{1}{\epsilon} \rceil + 1$ symbols in the d-ary encoding of N_1 to 0 to get $N_2 = b_{n-1}b_{n-2}\ldots b_{n-k}0\ldots 0$. We attempt to achieve a square of size $2N_2 + n$ (but will overshoot slightly). Let $N_3 = 1\underbrace{00\ldots 0}_{n} - N_2 = c_{n-1}c_{n-2}\ldots c_{n-k}\underbrace{00\ldots 0}_{n-k}$.

Our construction has four logical components, described below. The final shape obtained is the union of these components. For logical clarity, the reader can think of the assembly proceeding component by component, though parts of some components may begin assembling before other components assemble completely.

1. *L-shaped seed assembly*: Each tile in this component is unique and attaches to two adjacent tiles in the same component via strength 2 pads. The k tiles of the vertical arm of the L-shaped assembly sequentially encode the symbols $c_{n-1}, c_{n-2}, \ldots, c_{n-k}$, with c_{n-1} being the Northern most. The tiles of the horizontal arm encode a *seed row* for a base $m = \left\lceil \frac{\log(n-k)}{\log\log(n-k)} \right\rceil$ counter (called the *minor counter*) with vertical length $n-k$. Thus, there are $\Theta\left(\frac{\log(n-k)}{\log\log(n-k)}\right)$ tiles on the horizontal arm, which encode $e_{m-1}, e_{m-2}, \ldots, e_0$, East to West. The number of tile types used in this component is $\Theta\left(k + \frac{\log(n-k)}{\log\log(n-k)}\right)$.

2. *Minor counter*: This counter is seeded by the horizontal arm of the L-shaped assembly. It forms a rectangle of width m and height $n-k$ (excluding the *seed row*) by counting in base $\Theta\left\lceil \frac{\log(n-k)}{\log\log(n-k)} \right\rceil$. Note that the East pads of each tile that assembles on the Eastern border of the counter can be easily set to a unique special pad. We will use this pad to attach a unique tile encoding the symbol 0 (in base d) all along the Eastern border of the minor counter. The vertical arm of the L-shaped assembly and this vertical row of 0 tiles together represent the d-ary enoding of the number N_3. The number of tile types used in this component is $\Theta\left(\frac{\log(n-k)}{\log\log(n-k)}\right)$.

3. *Major counter*: The major counter uses the n length vertical *seed column* representing the d-ary encoding of N_3 to count upto $1\underbrace{00\ldots0}_{n}$ in base d to get an $n \times 2N_2$ rectangle (inclusive of the *seed column*). Note that the Northeastern most tile of the counter occurs exactly once in the assembly, and its North and East pads are not involved in the logic of the counter assembly. The number of tile types used in this component is $\Theta(d)$, since we count in base d.

4. *Diagonal and filler tiles*: Having constructed the major counter, we now complete the square. The bounded rectangular area to the North and West of the L-shaped seed assembly can be filled in using a single tile type whose North and South pads are identical to the North pad of the tiles forming the horizontal arm of the L-shaped seed assembly and West and East pads are identical to the West pad of the tiles forming the vertical arm of the L-shaped seed. Both the pads have strength one. From the northeastern most tile in the major counter, we will initiate an AB-type diagonal for the square (see section 4.2) in two directions, Northwest and Southeast, and then use filler tiles to complete the square. We require pads on the Eastern boundary of the major counter that are distinct from any pads appearing within the major counter so that the filler tiles do not interfere with the major counter assembly. This is done by adding a single vertical column of tiles to the East of the Eastern boundary of the major counter. A unique distinct strength 2 East pad on the second from top tile in the vertical column initiates the diagonal in the Southeastern direction, while a unique distinct strength 2 North pad on the Northeastern most pad of the major counter initiates the diagonal in the Northeastern direction. The complete square has size $L = m + n + 2N_2$

   All the East pads on the vertical column (except the top two) are identical and match the corresponding pads on the filler tiles. All the North pads on the Northern boundary of the major counter are identical (except the Eastern most) and match the corresponding pads on the filler tiles. The vertical column to the East of the major counter can be achieved with just three tile types, two for the top two tiles and one for the rest. Both the parts of the AB diagonals require two tile types each. Two types of filler tile types are required, one fills the area above the diagonal and another fills the area below. Thus, the number of tile types used in this component is 10.

5.2 Analysis

We will first argue that our construction yields a unique terminal assembly in the shape of a square. Then we will show that an appropriate choice of the base d for the major counter gives us a construction of the claimed size and number of tile types. The first component, the L-shaped assembly, clearly uniquely assembles into the shape illustrated in Fig. 3. The minor counter and the major counter are implemented exactly as described in Fig. 2(iii), apart from some blunt pads on the boundary being changed to distinct pads that do not have any binding with any other pads involved in the assembly of the counter. The correctness of these counters have already been proved previously (see Rothemund and Winfree (2000); Adleman et al. (2001)). By using distinct pad types for the two counters, and the L-shaped seed assembly, we ensure they do not have any undesired interactions. Finally, the diagonal and filler tiles are few in number (only 10) and their correctness can be verified by the reader.

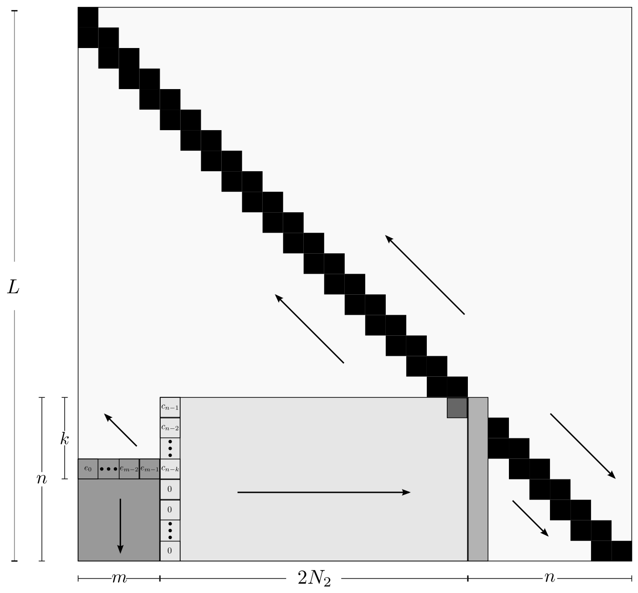

Figure 4: Diagonal and filler tiles complete the approximate square of length $L = 2N_2 + m + n$.

Lemma 1. *The construction described in section 5.1 produces a unique terminal $L \times L$ square.*

The choice of the base d affects the number of tile types in the vertical arm of the L-shaped seed assembly and also the number of tile types used in the major counter. A bigger value for d reduces the number of tile types used for the vertical arm, but increases the number of tile types in the major counter, while a smaller value does the opposite. We balance these two factors by choosing $d = \lceil \frac{\log \frac{1}{\epsilon}}{\log\log \frac{1}{\epsilon}} \rceil$. Note that this value, and hence the number of tile types for the major counter, is independent of N.

Theorem 1. *For the construction described above, for all $\epsilon \in (0, \frac{1}{4}]$ and sufficiently large N, a choice of $d = \left\lceil \frac{\log \frac{1}{\epsilon}}{\log\log \frac{1}{\epsilon}} \right\rceil$ produces a unique terminal $L \times L$ square where $(1-\epsilon)N \le L \le (1+\epsilon)N$ and uses $\mathcal{O}\left(\frac{\log \frac{1}{\epsilon}}{\log\log \frac{1}{\epsilon}} + \frac{\log\log \epsilon N}{\log\log\log \epsilon N}\right)$ tile types.*

Proof. Recall that $L = m + 2N_2 + n$; $d = \left\lceil \frac{\log \frac{1}{\epsilon}}{\log\log \frac{1}{\epsilon}} \right\rceil$; $N_1 = \left\lfloor \frac{N - \lfloor \log_d N \rfloor}{2} \right\rfloor$; $m = \left\lceil \frac{\log(n-k)}{\log\log(n-k)} \right\rceil$; $n = \lfloor \log_d N_1 \rfloor + 1$ and $k = \lceil \log_d \frac{1}{\epsilon} \rceil + 1$. Also, $n - k = \lfloor \log_d N_1 \rfloor + 1 - \lceil \log_d \frac{1}{\epsilon} \rceil - 1 \le \log_d(\epsilon N_1)$

Claim 1: For all $\epsilon \in (0, \frac{1}{4}]$ and all sufficiently large N, $L \le (1+\epsilon)N$.

We first upper bound n, m and $2N_2$:

1. $n = \lfloor \log_d N_1 \rfloor + 1 \le \log_d(N - \lfloor \log_d N \rfloor) - \log_d 2 + 1 \le \log_d(N - \lfloor \log_d N \rfloor) + 1 \le \log_d N + 1.$
2. $m = \left\lceil \frac{\log(n-k)}{\log\log(n-k)} \right\rceil \le 1 + \log(n-k) \le \log(\log_d(\epsilon N_1) + 1) + 1 \le 2\log\log_d(\epsilon N_1) \le 2\log\log_d(\epsilon N).$
3. $2N_2 \le 2N_1 \le N - \lfloor \log_d N \rfloor \le N - \log_d N + 1.$

So $L = m + 2N_2 + n \le 2\log\log_d(\epsilon N) + N - \log_d N + 1 + \log_d N + 1 = N + 2\log\log_d(\epsilon N) + 2 \le (1+\epsilon)N$ for sufficiently large N.

Claim 2: For all $\epsilon \in (0, \frac{1}{4}]$ and all sufficiently large N, $L \geq (1-\epsilon)N$.

Recall that $N_1 = b_{n-1}b_{n-2}\dots b_0$; $N_2 = b_{n-1}b_{n-2}\dots b_{n-k}\underbrace{0\dots0}_{n-k}$ and so $N_1 - N_2 = b_{n-k-1}b_{n-k-2}\dots b_0 \leq d^{n-k}-1 \leq \epsilon N_1 - 1$. Hence $2N_2 \geq 2N_1(1-\epsilon)+2$. Also, $n = \lfloor \log_d N_1 \rfloor + 1 \geq \log_d N_1 \geq \log_d\left(\frac{N-\lfloor \log_d N\rfloor - 2}{2}\right) = \log_d(N - \lfloor \log_d N \rfloor - 2) - \log_d 2 \geq \log_d \frac{N}{2} - \log_d 2 \geq \log_d N - 2 \geq \lfloor \log_d N \rfloor - 2$ and $2N_1 = 2\left\lfloor \frac{N-\lfloor \log_d N\rfloor}{2}\right\rfloor \geq N - \lfloor \log_d N \rfloor - 2$. Thus $L = m + 2N_2 + n \geq 2 + 2N_2 + n \geq 2 + 2N_1(1-\epsilon) + 2 + \lfloor \log_d N_1 \rfloor + 1 \geq 2 + 2N_1(1-\epsilon) + 2 + \lfloor \log_d N \rfloor - 2 \geq 2 + 2N_1 - 2\epsilon N_1 + \lfloor \log_d N \rfloor \geq N - 2\epsilon N_1 \geq N - \epsilon N = (1-\epsilon)N$ for sufficiently large N.

Claim 3: The number of tile types used in the construction is $\mathcal{O}\left(\frac{\log\frac{1}{\epsilon}}{\log\log\frac{1}{\epsilon}} + \frac{\log\log \epsilon N}{\log\log\log \epsilon N}\right)$.

The number of tile types used in the L-shaped seed assembly is $\Theta\left(k + \frac{\log(n-k)}{\log\log(n-k)}\right)$, the number of tile types used in the minor counter is $\Theta\left(\frac{\log(n-k)}{\log\log(n-k)}\right)$ and in the major counter is $\Theta(d)$. The diagonal and filler assemblies use 10 tile types. Thus the total number of tile types for our construction is $\mathcal{O}\left(\frac{\log\frac{1}{\epsilon}}{\log\log\frac{1}{\epsilon}} + \frac{\log\log \epsilon N}{\log\log\log \epsilon N}\right)$. □

5.3 Lower Bound

In this section we derive a lower bound on the number of tile types required to assemble an ϵ-approximation of an $N \times N$ square. We use information theoretic arguments and hence our bounds hold for almost all $N \in \mathbb{N}$. We are interested in approximation ratios less than $\frac{1}{4}$. In the following Lemma we relate the lower bound function obtained in Theorem 1 from information theoretic arguments to the upper bound from Theorem 1.

Lemma 2. *For all* $\epsilon \in (0, \frac{1}{4}]$: $\frac{\log(\frac{1}{\epsilon}+\log N)}{\log(\log\frac{1}{\epsilon}+\log\log \epsilon N)} = \Omega\left(\frac{\log\frac{1}{\epsilon}}{\log\log\frac{1}{\epsilon}} + \frac{\log\log \epsilon N}{\log\log\log \epsilon N}\right)$

Proof. We shall prove the claim separately for $\frac{1}{\epsilon} \leq \log N$ and $\frac{1}{\epsilon} \geq \log N$. For $\frac{1}{\epsilon} \leq \log N$, we have $\frac{\log\frac{1}{\epsilon}}{\log\log\frac{1}{\epsilon}} + \frac{\log\log\epsilon N}{\log\log\log\epsilon N} \leq \frac{\log\frac{1}{\epsilon}}{\log\log\frac{1}{\epsilon}} + \frac{\log\log N}{\log\log\log N} \leq \frac{2\log\log N}{\log\log\log N} \leq \frac{2\log(\frac{1}{\epsilon}+\log N)}{\log\log\log N} = \frac{4\log(\frac{1}{\epsilon}+\log N)}{2\log\log\log N} \leq \frac{4\log(\frac{1}{\epsilon}+\log N)}{\log(\log\log N+\log\frac{1}{\epsilon})} \leq 4\frac{\log(\frac{1}{\epsilon}+\log N)}{\log(\log\frac{1}{\epsilon}+\log\log\epsilon N)}$. Similarly for $\frac{1}{\epsilon} \geq \log N$, we have $\frac{\log\frac{1}{\epsilon}}{\log\log\frac{1}{\epsilon}} + \frac{\log\log\epsilon N}{\log\log\log\epsilon N} \leq \frac{\log\frac{1}{\epsilon}}{\log\log\frac{1}{\epsilon}} + \frac{\log\log N}{\log\log\log N} \leq \frac{2\log\frac{1}{\epsilon}}{\log\log\frac{1}{\epsilon}} \leq \frac{2\log(\frac{1}{\epsilon}+\log N)}{\log\log\frac{1}{\epsilon}} = \frac{4\log(\frac{1}{\epsilon}+\log N)}{2\log\log\frac{1}{\epsilon}} \leq \frac{4\log(\frac{1}{\epsilon}+\log N)}{\log(\log\log N+\log\frac{1}{\epsilon})} \leq 4\frac{\log(\frac{1}{\epsilon}+\log N)}{\log(\log\frac{1}{\epsilon}+\log\log\epsilon N)}$. □

Theorem 2. *For all* $\epsilon \in (0, \frac{1}{4})$*, the number of tile types required to assemble an* ϵ*-approximation of an* $N \times N$ *square is* $\Omega\left(\frac{\log\frac{1}{\epsilon}}{\log\log\frac{1}{\epsilon}} + \frac{\log\log\epsilon N}{\log\log\log\epsilon N}\right)$ *almost always.*

Proof. Let L be the size of an ϵ-approximation of an $N \times N$ square. Then we know that $(1-\epsilon)N \leq L \leq (1+\epsilon)N$ which implies $\frac{N}{2} \leq L \leq 2N$ since $\epsilon < \frac{1}{2}$. So the number of bits in the binary encoding of L, $n = \lfloor \log L \rfloor + 1 \geq \lfloor \log N \rfloor$. Any construction that builds an ϵ-approximation of an $N \times N$ square is an implicit encoding of the number n. Also, from L, we can recover the first $\lceil \log\frac{1}{\epsilon}\rceil$ bits of N. Let these bits represent the number $e \geq 2^{\lceil \log\frac{1}{\epsilon}\rceil - 1} \geq \frac{1}{2\epsilon}$. Thus, from the assembled square, we implicitly get the number $n' = n + e \geq \lfloor \log N \rfloor + \frac{1}{2\epsilon}$. Thus, by arguments similar to Rothemund and Winfree (2000), we have $(1-\delta)\log n' < \mathcal{K}_U(n') \leq |p_{SA}| + f(K_\epsilon(N))$ for any $0 < \delta < 1$ and almost all N. Here, $\mathcal{K}_U(n')$ is the Kolmogorov complexity of the binary number n' with respect to the universal Turing machine U, $K_\epsilon(N)$ is the minimum number of tile types to form an ϵ-approximation of an $N \times N$ square, p_{SA} is a program that on input a tile set for an $N \times N$ ϵ-approximate square outputs n' and $f(K_\epsilon(N)) = \mathcal{O}(K_\epsilon(N)\log(K_\epsilon(N)))$ is the size of the encoding of the tile set of size $K_\epsilon(N)$. Thus, $(1-\delta)\log\left(\lfloor \log N \rfloor + \frac{1}{2\epsilon}\right) \leq c_1 + c_2 K_\epsilon(N)\log(K_\epsilon(N))$. From Theorem 1, $K_\epsilon(N) \leq \mathcal{O}\left(\frac{\log\frac{1}{\epsilon}}{\log\log\frac{1}{\epsilon}} + \frac{\log\log\epsilon N}{\log\log\log\epsilon N}\right)$ and so $(1-\delta)\log\left(\lfloor \log N \rfloor + \frac{1}{2\epsilon}\right) \leq c_1 + c_2 K_\epsilon(N)\log\left(\frac{\log\frac{1}{\epsilon}}{\log\log\frac{1}{\epsilon}} + \frac{\log\log\epsilon N}{\log\log\log\epsilon N}\right)$. Simplifying, we get $K_\epsilon(N) = \Omega\left(\frac{\log(\frac{1}{\epsilon}+\log N)}{\log(\log\frac{1}{\epsilon}+\log\log\epsilon N)}\right)$ almost always and hence by Lemma 2, $K_\epsilon(N) = \Omega\left(\frac{\log\frac{1}{\epsilon}}{\log\log\frac{1}{\epsilon}} + \frac{\log\log\epsilon N}{\log\log\log\epsilon N}\right)$ almost always. □

6 Lower Bounds on Tile Complexity of Arbitrary Shapes

In this section we show two lower bounds for minimum tile set of arbitrary shapes by using Kolmogorov complexity techniques like the one introduced by Rothemund and Winfree (2000). We associate each shape with a binary number via a deterministic, computable procedure. Each shape is then an encoding of a binary number and there exists a fixed size program that on input a tile set for that shape outputs the associated binary number. The description of the tile set together with the description of this program cannot be smaller than the Kolmogorov complexity of the binary number and this provides a lower bound on the minimum tile set for assembling the given shape.

Theorem 3. *Almost always, any shape with N tiles requires $\Omega(\frac{\log N}{\log\log N})$ tile types for its assembly.*

Proof. Given a universal Turing Machine U, there exists a program p_{count} which on input a valid self-assembly program in the form of a tile set T simulates the assembly and outputs the number of tiles in the terminal assembly if it halts. Such a program, along with the tile set T is an encoding of N, the number of tiles in the terminal assembly of T. Thus by arguments similar to previous section, $(1-\delta)\log N < \mathcal{K}_U(N) \leq |p_{count}| + c_1 K_{shape}(c_2 + \log K_{shape})$ where K_{shape} is the smallest tile set that produces the given shape. Simplifying gives the required lower bound. □

In Theorem 4 below, we describe a method to decompose any arbitrary shape into multiple non-overlapping squares. We call this the *maximal squares decomposition* and call each square in the decomposition a *maximal square*. This decomposition is achieved by repeatedly choosing the largest side square lying entirely within the shape and deleting it. Each square thus chosen is a maximal square. In each round, if more than one candidate maximal square has the same length side, we unambiguously choose the square whose North-East extent co-ordinates are greatest. This procedure might disconnect the shape in which case we independently apply our procedure for each component recursively in a breadth first manner. The procedure is guaranteed to halt since at least one tile is deleted in each round.

Theorem 4. *Almost always, any shape requires at least $\Omega(\frac{N}{\log N})$ tile types for its assembly where N is the size of the largest maximal square embedded in the shape.*

Proof. For any given arbitrary shape S, let $\{n_1, n_2, \ldots, n_k\}$ be the sides of the maximal squares of its maximal square decomposition ordered decreasingly. Ordering between maximal squares of the same side length is immaterial. Let N be the size of the largest maximal square embedded in S and so $N = n_1$. Let $L = \sum_{i=1}^{k} 2^{n_i - 1}$ and so $N - 1 \leq \log L$. Given a universal Turing Machine U, there exists a program p_{maxsq} which on input a valid self-assembly program in form of a tile set T simulates the assembly and outputs L if the assembly halts. Such a program, along with the tile set T is an encoding of L. Thus by arguments similar to previous section, $(1-\delta)(N-1) \leq (1-\delta)\log L < \mathcal{K}_U(L) \leq |p_{maxsq}| + c_1 K_{shape}(c_2 + \log K_{shape})$ where K_{shape} is the smallest tile set that produces the given shape. Simplifying gives the required lower bound. □

7 Discussion and Open Problems

The assembly time (as defined in Adleman et al. (2001)) of our construction can be improved without affecting the asymptotic tile complexity by using fast parallel binary counters instead of our current higher base counter. This can be achieved using simple base converters described by Adleman et al. (2001). Most of the tiles that make up each approximately sized square are either filler tiles, diagonal tiles or counter tiles. It is useful for these sub-assemblies to be constructed from a fixed set of tiles that do not depend on ϵ and N. There are 10 fixed diagonal and filler tile types and we can use 10 fixed tile types for each of the two parallel binary counters and hence the major portion of any approximate square can be constructed from this fixed set of 30 tile types which can be synthesized in bulk. Laboratory implementation of simple constructions from the Tile Assembly Model using DNA have been successful carried out (Rothemund et al.

(2004)). However these implementations have been beset by errors due to spurious nucleation and coordinated binding errors. Various error correction and prevention techniques have been studied to deal with these challenges (see Winfree and Bekbolatov (2003); Chen and Goel (2004); Schulman and Winfree (2009); Reif et al. (2004)). Our construction suffers from these same kind of errors and the same error correcting techniques proposed previously apply to our constructions. For any given ϵ, our constructions have exponentially smaller tile sets than the optimal tile sets for exact squares for all large enough N. Such large constructions are not currently achievable in the laboratory, though the field is progressing rapidly. Our notion of approximation led us to construct target shapes that were approximately sized but in the shape of squares. Under different notions of approximation, one could analyze similar questions. Another challenge is to develop notions of approximation for arbitrary shapes, constructing these and lower bounding their tile complexity in TAM. The general technique illustrated in section 6 may be used to supply other lower bounds for either arbitrary or special classes of shapes by finding different methods for encoding shapes as binary numbers.

References

Adleman, L. (2000). Towards a Mathematical Theory of Self-assembly. Technical report, University of Southern California.

Adleman, L., Cheng, Q., Goel, A., and Huang, M.-D. (2001). Running Time and Program Size for Self-Assembled Squares. *Symposium on Theory of Computing*, pages 740–748.

Adleman, L., Cheng, Q., Goel, A., Huang, M.-D., Kempe, D., de Espanes, P. M., and Rothemund, P. (2002). Combinatorial Optimization Problems in Self-Assembly. *Symposium on Theory of Computing*, pages 23–32.

Aggarwal, G., Goldwasser, M., Kao, M.-Y., and Schweller, R. (2004). Complexities for Generalized Models of Self-Assembly. *Symposium on Discrete Algorithms*, pages 880–889.

Andersen, E., Dong, M., Nielsen, M., Jahn, K., Subramani, R., Mamdouh, W., Golas, M., Sander, B., Stark, H., Oliveira, C., Pedersen, J. S., Birkedal, V., Besenbacher, F., Gothelf, K., and Kjems, J. (2009). Self-assembly of a Nanoscale DNA Box with a Controllable Lid. *Nature*, 459(7243):73–76.

Barish, R., Rothemund, P., and Winfree, E. (2005). Two Computational Primitives for Algorithmic Self-Assembly: Copying and Counting. *Nano Letters*, 5:2586–2592.

Becker, F., Rapaport, I., and Remila, E. (2006). Self-assemblying Classes of Shapes with a Minimum Number of Tiles, and in Optimal Time. *Foundations of Software Technology and Theoretical Computer Science*, pages 45–56.

Berger, R. (1966). The Undecidability of the Domino Problem. *Memoirs of American Mathematical Society*, 66:1–72.

Chandran, H., Gopalkrishnan, N., and Reif, J. (2009). The Tile Complexity of Linear Assemblies. *International Colloquium on Automata, Languages and Programming*, pages 235–253.

Chen, H.-L. and Goel, A. (2004). Error Free Self-assembly Using Error Prone Tiles. *DNA Computing*, pages 62–75.

Demaine, E., Demaine, M., Fekete, S., Ishaque, M., Rafalin, E., Schweller, R., and Souvaine, D. (2007). Staged Self-assembly: Nanomanufacture of Arbitrary Shapes with $O(1)$ Glues. *DNA Computing*, pages 1–14.

Dietz, H., Douglas, S., and Shih, W. (2009). Folding DNA into Twisted and Curved Nanoscale Shapes. *Science*, 325(5941):725–730.

Dirks, R. and Pierce, N. (2004). Triggered Amplification by Hybridization Chain Reaction. *Proceedings of the National Academy of Sciences of the United States of America*, 101(43):15275–15278.

Doty, D. (2009). Randomized Self-Assembly for Exact Shapes. *Foundations of Computer Science*.

Doty, D., Patitz, M., and Summers, S. (2009). Limitations of Self-Assembly at Temperature 1. *DNA Computing*, 5877:35–44.

Douglas, S., Dietz, H., Liedl, T., Hogberg, B., Graf, F., and Shih, W. (2009). Self-assembly of

DNA into Nanoscale Three-dimensional Shapes. *Nature*, 459(7245):414–418.

Garey, M. and Johnson, D. (1981). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.

Kao, M.-Y. and Schweller, R. (2006). Reducing Tile Complexity for Self-assembly through Temperature Programming. *Symposium on Discrete Algorithms*, pages 571–580.

Kao, M.-Y. and Schweller, R. (2008). Randomized Self-Assembly for Approximate Shapes. *International Colloquium on Automata, Languages and Programming*, pages 370–384.

LaBean, T., Yan, H., Kopatsch, J., Liu, F., Winfree, E., Reif, J., and Seeman, N. (2000). Construction, Analysis, Ligation, and Self-Assembly of DNA Triple Crossover Complexes. *Journal of the American Chemical Society*, 122(9):1848–1860.

Lewis, H. and Papadimitriou, C. (1981). *Elements of the Theory of Computation*. Prentice Hall.

Mao, C., Labean, T., Reif, J., and Seeman, N. (2000). Logical Computation Using Algorithmic Self-assembly of DNA Triple-crossover Molecules. *Nature*, 407:493–496.

Park, S.-H., Yin, P., Liu, Y., Reif, J., LaBean, T., and Yan, H. (2005). Programmable DNA Self-assemblies for Nanoscale Organization of Ligands and Proteins. *Nano Letters*, 5:729–733.

Reif, J., Sahu, S., and Yin, P. (2004). Compact Error-Resilient Computational DNA Tiling Assemblies. *DNA Computing*, pages 293–307.

Robinson, R. (1971). Undecidability and Nonperiodicity for Tilings of the Plane. *Inventiones Mathematicae*, 12:177–209.

Rothemund, P. (2006). Folding DNA to Create Nanoscale Shapes and Patterns. *Nature*, 440:297–302.

Rothemund, P., Papadakis, N., and Winfree, E. (2004). Algorithmic Self-Assembly of DNA Sierpinski Triangles. *PLoS Biology*, 2.

Rothemund, P. and Winfree, E. (2000). The Program-Size Complexity of Self-Assembled Squares. *Symposium on Theory of Computing*, pages 459–468.

Schulman, R. and Winfree, E. (2009). Programmable Control of Nucleation for Algorithmic Self-Assembly. *SIAM Journal on Computing*, 39(4):1581–1616.

Soloveichik, D. and Winfree, E. (2007). Complexity of Self-Assembled Shapes. *SIAM Journal of Computing*, 36:1544–1569.

Wang, H. (1961). Proving Theorems by Pattern Recognition II. *Bell Systems Technical Journal*.

Winfree, E. (1995). On the Computational Power of DNA Annealing and Ligation. *DNA Based Computers,DIMACS*.

Winfree, E. and Bekbolatov, R. (2003). Proofreading Tile Sets: Error Correction for Algorithmic Self-Assembly. *DNA Computing*, pages 126–144.

Winfree, E., Liu, F., Wenzler, L., and Seeman, N. (1998). Design and Self-assembly of Two-dimensional DNA Crystals. *Nature*, 394:539–544.

Winfree, E., Yang, X., and Seeman, N. (1996). Universal Computation via Self-assembly of DNA: Some Theory and Experiments. *DNA Based Computers II, DIMACS*, 44:191–213.

Yan, H., Feng, L., LaBean, T., and Reif, J. (2003). Parallel Molecular Computation of Pair-Wise XOR using DNA String Tile. *Journal of the American Chemical Society*, 125.

Yin, P., Choi, H., Calvert, C., and Pierce, N. (2008). Programming Biomolecular Self-assembly Pathways. *Nature*, 451(7176):318–322.

Yin, P., Yan, H., Daniell, X., Turberfield, A., and Reif, J. (2004). A Unidirectional DNA Walker Moving Autonomously Along a Linear Track. *Angewandte Chemie International Edition*, 116(37):5014–5019.

Zhang, D., Turberfield, A., Yurke, B., and Winfree, E. (2007). Engineering Entropy-Driven Reactions and Networks Catalyzed by DNA. *Science*, 318:1121–1125.